Assignment 2.1
Module 2

Name: Shikha Singh
Roll no: 25

Q1. WAP to create a class called Circle. It contains:
• private instance variable: radius (of the type double) with default value of 1.0.
• Two overloaded constructors - a default constructor with no argument, and a constructor which takes a double argument for radius.
• Two public methods: getRadius(), calculateArea(), calculateCircumference() which return the radius, calculate and return area, and circumference respectively.
Hint: Use Math.PI for calculating area and circumference

```java
import java.util.Scanner;

public class Circle {

   private double radius;


   public Circle() {
       this.radius = 1.0;
   }


   public Circle(double radius) {
       this.radius = radius;
   }


   public double getRadius() {
       return this.radius;
   }


   public double calculateArea() {
       return Math.PI * radius * radius;
   }
```

```
    public double calculateCircumference() {
        return 2 * Math.PI * radius;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);


        System.out.print("Enter the radius of the circle: ");
        double inputRadius = scanner.nextDouble();


        Circle circle = new Circle(inputRadius);


        System.out.println("Radius: " + circle.getRadius());
        System.out.println("Area: " + circle.calculateArea());
        System.out.println("Circumference: " + circle.calculateCircumference());


        scanner.close();
    }
}
```

```
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 % cd "
ikhasingh/Desktop/java assignment 2.1/" && javac Circle.ja
a Circle
Enter the radius of the circle: 23
Radius: 23.0
Area: 1661.9025137490005
Circumference: 144.51326206513048
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
```

Q2. A class called Rectangle, which models a rectangle with a length and a width (in float), is designed as shown in the following class diagram. Write the Rectangle class as per UML diagram.

```
public class Rectangle {
```

```java
private float length = 1.0f;
private float width = 1.0f;

public Rectangle() {

}


public Rectangle(float length, float width) {
    this.length = length;
    this.width = width;
}


public float getLength() {
    return length;
}


public void setLength(float length) {
    this.length = length;
}


public float getWidth() {
    return width;
}

public void setWidth(float width) {
    this.width = width;
}

public double getArea() {
    return length * width;
}


public double getPerimeter() {
    return 2 * (length + width);
}
```

```
    public String toString() {
        return "Rectangle[length=" + length + ",width=" + width + "]";
    }


    public static void main(String[] args) {
        java.util.Scanner scanner = new java.util.Scanner(System.in);


        System.out.print("Enter the length of the rectangle: ");
        float userLength = scanner.nextFloat();

        System.out.print("Enter the width of the rectangle: ");
        float userWidth = scanner.nextFloat();


        Rectangle rectangle = new Rectangle(userLength, userWidth);


        System.out.println("\nRectangle Details:");
        System.out.println("Length: " + rectangle.getLength());
        System.out.println("Width: " + rectangle.getWidth());
        System.out.println("Area: " + rectangle.getArea());
        System.out.println("Perimeter: " + rectangle.getPerimeter());
        System.out.println("String Representation: " + rectangle.toString());


        scanner.close();
    }
}
```

```
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 % cd "/Users/sh
ikhasingh/Desktop/java assignment 2.1/" && javac Rectangle.java &&
java Rectangle
Enter the length of the rectangle: 4
Enter the width of the rectangle: 5

Rectangle Details:
Length: 4.0
Width: 5.0
Area: 20.0
Perimeter: 18.0
String Representation: Rectangle[length=4.0,width=5.0]
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
```

Q3. A class called Employee, which models an employee with an ID, name and salary, is designed
as shown in the following class diagram. The method raiseSalary(percent) increases the salary by
the given percentage. Write the Employee class and its driver class.

```java
public class Employee {

   private int id;
   private String firstName;
   private String lastName;
   private int salary;

   public Employee(int id, String firstName, String lastName, int salary) {
       this.id = id;
       this.firstName = firstName;
       this.lastName = lastName;
       this.salary = salary;
   }



   public int getId() {
       return id;
   }



   public String getFirstName() {
       return firstName;
   }
```

```java
public String getLastName() {
    return lastName;
}

public String getName() {
    return firstName + " " + lastName;
}


public int getSalary() {
    return salary;
}


public void setSalary(int salary) {
    this.salary = salary;
}


public int getAnnualSalary() {
    return salary * 12;
}


public int raiseSalary(int percent) {
    salary += salary * percent / 100;
    return salary;
}


public String toString() {
    return "Employee[id=" + id + ",name=" + getName() + ",salary=" + salary + "]";
}

public static void main(String[] args) {
    java.util.Scanner scanner = new java.util.Scanner(System.in);

    System.out.print("Enter employee ID: ");
    int userId = scanner.nextInt();
```

```
        System.out.print("Enter first name: ");
        String userFirstName = scanner.next();

        System.out.print("Enter last name: ");
        String userLastName = scanner.next();

        System.out.print("Enter salary: ");
        int userSalary = scanner.nextInt();

        Employee employee = new Employee(userId, userFirstName, userLastName,
userSalary);

        System.out.println("\nEmployee Details:");
        System.out.println("ID: " + employee.getId());
        System.out.println("Name: " + employee.getName());
        System.out.println("Salary: " + employee.getSalary());
        System.out.println("Annual Salary: " + employee.getAnnualSalary());


        System.out.print("\nEnter percentage to raise salary: ");
        int percent = scanner.nextInt();


        int newSalary = employee.raiseSalary(percent);
        System.out.println("New Salary after " + percent + "% raise: " + newSalary);


        System.out.println("String Representation: " + employee.toString());


        scanner.close();
    }
}
```

```
ikhasingh/Desktop/java assignment 2.1/" && javac Employee.java && j
ava Employee
Enter employee ID: 12
Enter first name: shikha
Enter last name: singh
Enter salary: 23000

Employee Details:
ID: 12
Name: shikha singh
Salary: 23000
Annual Salary: 276000

Enter percentage to raise salary: 3
New Salary after 3% raise: 23690
String Representation: Employee[id=12,name=shikha singh,salary=23690]
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
```

Q4. A class called InvoiceItem, which models an item of an invoice, with ID, description, quantity and unit price, is designed as shown in the following class diagram. It has a method getTotal which
calculates total value (total=quantity*unit price). Write the InvoiceItem class and it's driver class.

```java
public class InvoiceItem {

   private String id;
   private String desc;
   private int qty;
   private double unitPrice;



   public InvoiceItem(String id, String desc, int qty, double unitPrice) {
       this.id = id;
       this.desc = desc;
       this.qty = qty;
       this.unitPrice = unitPrice;
   }



   public String getId() {
       return id;
   }

   public String getDesc() {
       return desc;
```

```
    }


    public int getQty() {
        return qty;
    }

     public void setQty(int qty) {
        this.qty = qty;
    }


    public double getUnitPrice() {
        return unitPrice;
    }

    public void setUnitPrice(double unitPrice) {
        this.unitPrice = unitPrice;
    }


    public double getTotal() {
        return unitPrice * qty;
    }


    public String toString() {
        return "InvoiceItem[id=" + id + ",desc=" + desc + ",qty=" + qty + ",unitPrice="
+ unitPrice + "]";
    }


    public static void main(String[] args) {
        java.util.Scanner scanner = new java.util.Scanner(System.in);


        System.out.print("Enter item ID: ");
        String userId = scanner.nextLine();

        System.out.print("Enter item description: ");
        String userDesc = scanner.nextLine();
```

```
        System.out.print("Enter quantity: ");
        int userQty = scanner.nextInt();

        System.out.print("Enter unit price: ");
        double userUnitPrice = scanner.nextDouble();



        InvoiceItem item = new InvoiceItem(userId, userDesc, userQty, userUnitPrice);



        System.out.println("\nInvoice Item Details:");
        System.out.println("ID: " + item.getId());
        System.out.println("Description: " + item.getDesc());
        System.out.println("Quantity: " + item.getQty());
        System.out.println("Unit Price: " + item.getUnitPrice());
        System.out.println("Total Price: " + item.getTotal());



        System.out.println("String Representation: " + item.toString());

        scanner.close();
    }
}
```

```
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 % cd "/Users/shikhasingh/Desktop/ja
oiceItem.java && java InvoiceItem
Enter item ID: 2
Enter item description: this is good
Enter quantity: 3
Enter unit price: 23

Invoice Item Details:
ID: 2
Description: this is good
Quantity: 3
Unit Price: 23.0
Total Price: 69.0
String Representation: InvoiceItem[id=2,desc=this is good,qty=3,unitPrice=23.0]
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
```

Q5. A class called Author (as shown in the class diagram) is designed to model a book's author. It
contains:
• Three private instance variables: name (String), email (String), and gender (char of either 'm' or 'f');

• One constructor to initialize the name, email and gender with the given values; public Author (String name, String email, char gender) {.....}
(There is no default constructor for Author, as there are no defaults for name, email and gender.)
• public getters/setters: getName(), getEmail(), setEmail(), and getGender();
(There are no setters for name and gender, as these attributes cannot be changed.)
• A toString() method that returns "Author[name=?,email=?,gender=?]", e.g., “Author[name=Abc ,email=Abc@gmail.com, gender=m]".

```
import java.util.Scanner;

public class Author {

   private String name;
   private String email;
   private char gender;
   public Author(String name, String email, char gender) {
       this.name = name;
       this.email = email;
       this.gender = gender;
   }


   public String getName() {
       return name;
   }


   public String getEmail() {
       return email;
   }


   public void setEmail(String email) {
       this.email = email;
   }


   public char getGender() {
       return gender;
   }
```

```
    public String toString() {
        return "Author[name=" + name + ",email=" + email + ",gender=" + gender + "]";
    }


    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);


        System.out.print("Enter author's name: ");
        String name = scanner.nextLine();


        System.out.print("Enter author's email: ");
        String email = scanner.nextLine();


        System.out.print("Enter author's gender (m/f): ");
        char gender = scanner.next().charAt(0);


        Author author = new Author(name, email, gender);


        System.out.println("\nAuthor Details:");
        System.out.println("Name: " + author.getName());
        System.out.println("Email: " + author.getEmail());
        System.out.println("Gender: " + author.getGender());


        System.out.println("String Representation: " + author.toString());


        scanner.close();
    }
}
```

```
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 % cd "/Users/shikhasingh/Desktop
hor.java && java Author
Enter author's name: shikha
Enter author's email: shikha@1805gmail.com
Enter author's gender (m/f): f

Author Details:
Name: shikha
Email: shikha@1805gmail.com
Gender: f
String Representation: Author[name=shikha,email=shikha@1805gmail.com,gender=f]
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
```

Q6. WAP to create a class time having default constructor, parameterized constructor whose specifications are as follows:

• Instance variable: hr, min, sec

• Constructors:

▪ default (with no parameters passed; should initialize the represented time to 12:0:0)

▪ a constructor with three parameters: hours, minutes, and seconds.

▪ a constructor with one parameter: the value of time in seconds since midnight (it should be converted into the time value in hours, minutes, and seconds)

• Instance methods:

▪ setClock() with one parameter seconds since midnight (to be converted into the time value in hours, minutes, and seconds as above).

▪ tick() with no parameters that increments the time stored in a Clock object by one second.

▪ tickDown() which decrements the time stored in a Clock object by one second.

▪ displaytime() displays the time in the format hr: min:sec e.g: 05:45:23

```java
public class Time {

   private int hr;
   private int min;
   private int sec;

    public Time() {
        this.hr = 12;
        this.min = 0;
        this.sec = 0;
   }

   public Time(int hr, int min, int sec) {
        this.hr = hr;
        this.min = min;
        this.sec = sec;
```

```
}


public Time(int secondsSinceMidnight) {
    setClock(secondsSinceMidnight);
}


public void setClock(int secondsSinceMidnight) {
    this.hr = (secondsSinceMidnight / 3600) % 24;
    this.min = (secondsSinceMidnight / 60) % 60;
    this.sec = secondsSinceMidnight % 60;
}


public void tick() {
    setClock(hr * 3600 + min * 60 + sec + 1);
}


public void tickDown() {
    setClock(hr * 3600 + min * 60 + sec - 1);
}


public void displayTime() {

    System.out.printf("%02d:%02d:%02d\n", hr, min, sec);
}


public static void main(String[] args) {

    Time t1 = new Time();
    Time t2 = new Time(5, 45, 23);
    Time t3 = new Time(3723);


    System.out.println("Initial Times:");
    t1.displayTime();
    t2.displayTime();
    t3.displayTime();
```

```
        t1.tick();
        t2.tick();
        t3.tick();


        System.out.println("Times after ticking:");
        t1.displayTime();
        t2.displayTime();
        t3.displayTime();


        t1.tickDown();
        t2.tickDown();
        t3.tickDown();


        System.out.println("Times after ticking down:");
        t1.displayTime();
        t2.displayTime();
        t3.displayTime();
    }
}
```

```
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 % cd "/Users/shikhasingh/Desktop/java assignment 2.1/" && j
e.java && java Time
Initial Times:
12:00:00
05:45:23
01:02:03
Times after ticking:
12:00:01
05:45:24
01:02:04
Times after ticking down:
12:00:00
05:45:23
01:02:03
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
```

Q7. Write a Java class Complex for dealing with complex number. Your class must have the following features:

• Instance variables :

o real for the real part of type double
o imag for imaginary part of type double.

• Constructor:

o public Complex (): A default constructor, it should initialize the number to 0, 0)
o public Complex (double real, double imag: A constructor with parameters, it creates the complex object by setting the two fields to the passed values.

• Instance methods:

o public Complex add (Complex n): This method will find the sum of the current complex number and the passed complex number. The methods returns a new Complex number which is the sum of the two.

o public Complex subtract (Complex n): This method will find the difference of the current complex number and the passed complex number. The methods returns a new Complex number which is the difference of the two.
o public void setReal(double real): Used to set the real part of this complex number.
o public void setImag(double image): Used to set the imaginary part of this complex number.
o public double getReal(): This method returns the real part of the complex number
o public double getImag(): This method returns the imaginary part of the complex number
o public String toString(): This method allows the complex number to be easily printed out to the screen

Write a separate class ComplexDemo with a main() method and test the Complex class methods.

```
public class ComplexDemo {
   public static void main(String[] args) {

       Complex c1 = new Complex();
       Complex c2 = new Complex(3, 4);



       System.out.println("Initial Complex Numbers:");
       System.out.println("c1: " + c1);
       System.out.println("c2: " + c2);



       c1.setReal(1.5);
       c1.setImag(2.5);
```

```
        System.out.println("\nUpdated c1:");
        System.out.println("c1: " + c1);


        Complex sum = c1.add(c2);
        System.out.println("\nSum of c1 and c2:");
        System.out.println("c1 + c2 = " + sum);

        Complex diff = c1.subtract(c2);
        System.out.println("\nDifference of c1 and c2:");
        System.out.println("c1 - c2 = " + diff);
    }
}
class Complex {

    private double real;
    private double imag;


    public Complex() {
        this.real = 0;
        this.imag = 0;
    }


    public Complex(double real, double imag) {
        this.real = real;
        this.imag = imag;
    }


    public Complex add(Complex n) {
        double newReal = this.real + n.real;
        double newImag = this.imag + n.imag;
        return new Complex(newReal, newImag);
    }


    public Complex subtract(Complex n) {
        double newReal = this.real - n.real;
```

```
        double newImag = this.imag - n.imag;
        return new Complex(newReal, newImag);
    }


    public void setReal(double real) {
        this.real = real;
    }


    public void setImag(double imag) {
        this.imag = imag;
    }


    public double getReal() {
        return real;
    }


    public double getImag() {
        return imag;
    }


    public String toString() {
        return real + " + " + imag + "i";
    }
}
```

```
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 % c
plexDemo.java && java ComplexDemo
Initial Complex Numbers:
c1: 0.0 + 0.0i
c2: 3.0 + 4.0i

Updated c1:
c1: 1.5 + 2.5i

Sum of c1 and c2:
c1 + c2 = 4.5 + 6.5i

Difference of c1 and c2:
c1 - c2 = -1.5 + -1.5i
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
```

Q8. Create a Geometry class and overload calculateArea method for square, circle, and rectangle.

```java
class Geometry {



    public double calculateArea(double side) {
        return side * side;
    }



    public double calculateArea(double radius, boolean isCircle) {
        if (isCircle) {
            return Math.PI * radius * radius;
        }
        return 0;
    }



    public double calculateArea(double length, double width) {
        return length * width;
    }
}
public class GeometryDemo {
    public static void main(String[] args) {
```

```
        Geometry geometry = new Geometry();


        double squareArea = geometry.calculateArea(5);
        System.out.println("Area of the square: " + squareArea);

        double circleArea = geometry.calculateArea(7, true);
        System.out.println("Area of the circle: " + circleArea);


        double rectangleArea = geometry.calculateArea(4, 8);
        System.out.println("Area of the rectangle: " + rectangleArea);
    }
}
```

```
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 % cd "/Users/sh
metryDemo.java && java GeometryDemo
Area of the square: 25.0
Area of the circle: 153.93804002589985
Area of the rectangle: 32.0
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
```

Q9. WAP to implement Box class (Hint: Refer Java PPT). Inherit Box class in BoxWt whose

a. instance variable is weight and
b. method is print_BoxWt()
c. constructors: default, parameterized and BoxWt(BoxWt ob)

Use super() to invoke superclass constructors.

```
class Box {
    double width, height, depth;


    Box() {
        width = height = depth = 0;
    }


    Box(double w, double h, double d) {
        width = w;
```

```
        height = h;
        depth = d;
    }


    Box(Box ob) {
        width = ob.width;
        height = ob.height;
        depth = ob.depth;
    }


    double volume() {
        return width * height * depth;
    }


    void print_Box() {
        System.out.println("Dimensions are: " + width + " x " + height + " x " +
depth);
    }
}

class BoxWt extends Box {
    double weight;

     BoxWt() {
        super();
        weight = 0;
    }


    BoxWt(double w, double h, double d, double m) {
        super(w, h, d);
        weight = m;
    }

    BoxWt(BoxWt ob) {
        super(ob);
        weight = ob.weight;
    }
```

```
   void print_BoxWt() {
       print_Box();
       System.out.println("Weight: " + weight);
   }
}


public class  BoxDemo {
   public static void main(String[] args) {

       BoxWt bw1 = new BoxWt();
       BoxWt bw2 = new BoxWt(3, 5, 7, 8.37);
       BoxWt bw3 = new BoxWt(bw2);


       System.out.println("BoxWt 1:");
       bw1.print_BoxWt();

       System.out.println("\nBoxWt 2:");
       bw2.print_BoxWt();

       System.out.println("\nBoxWt 3 (Copy of BoxWt 2):");
       bw3.print_BoxWt();
   }
}
```

```
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 % cd "/Users/shikhasing
Demo.java && java BoxDemo
BoxWt 1:
Dimensions are: 0.0 x 0.0 x 0.0
Weight: 0.0

BoxWt 2:
Dimensions are: 3.0 x 5.0 x 7.0
Weight: 8.37

BoxWt 3 (Copy of BoxWt 2):
Dimensions are: 3.0 x 5.0 x 7.0
Weight: 8.37
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
```

Q10. WAP the demonstrate that super class variable can point to object of subclass.

```
// Superclass
```

```java
class Animal {
   void sound() {
       System.out.println("Animal makes a sound");
   }
}

// Subclass
class Dog extends Animal {

   void sound() {
       System.out.println("Dog barks");
   }
}



public class Super {
   public static void main(String[] args) {

       Animal myAnimal = new Dog();









       Animal[] animals = {new Animal(), new Dog()};

       for (Animal animal : animals) {
           animal.sound();
       }
   }
}
```

```
shikhasingh@SHIKHAs-MacBook-Air java a
er.java && java Super
Animal makes a sound
Dog barks
shikhasingh@SHIKHAs-MacBook-Air java a
```

Q11.WAP to demonstrate multilevel inheritance by creating a BoxColor class and inheriting BoxWt
class. BoxColor class has an instance variable color of String type.

```
class Box {
   double width, height, depth;

   Box() {
       width = height = depth = 0;
   }

   Box(double w, double h, double d) {
       width = w;
       height = h;
       depth = d;
   }

   Box(Box ob) {
       width = ob.width;
       height = ob.height;
       depth = ob.depth;
   }

   double volume() {
       return width * height * depth;
   }

   void print_Box() {
       System.out.println("Dimensions are: " + width + " x " + height + " x " +
depth);
   }
}

class BoxWt extends Box {
   double weight;

   BoxWt() {
       super();
       weight = 0;
   }

   BoxWt(double w, double h, double d, double m) {
```

```
        super(w, h, d);
        weight = m;
    }

    BoxWt(BoxWt ob) {
        super(ob);
        weight = ob.weight;
    }

    void print_BoxWt() {
        print_Box();
        System.out.println("Weight: " + weight);
    }
}

class BoxColor extends BoxWt {
    String color;

    BoxColor() {
        super();
        color = "undefined";
    }

    BoxColor(double w, double h, double d, double m, String c) {
        super(w, h, d, m);
        color = c;
    }

    BoxColor(BoxColor ob) {
        super(ob);
        color = ob.color;
    }

    void print_BoxColor() {
        print_BoxWt();
        System.out.println("Color: " + color);
    }
}

class BoxColorTest {
    public static void main(String[] args) {
        BoxColor bc = new BoxColor(3, 5, 7, 8.37, "Red");
```

```
        bc.print_BoxColor();
    }
}
```

```
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 % cd "/Users/s
ColorTest.java && java BoxColorTest
Dimensions are: 3.0 x 5.0 x 7.0
Weight: 8.37
Color: Red
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
```

Q12.WAP to demonstrate the sequence of execution of constructors in multilevel inheritance.

```
interface Figure {
    double PI = 3.14159;

    double area();
    double perimeter();
}

interface Draw {
    void draw_shape();
}

class Circle implements Figure, Draw {
    double radius;

    Circle(double radius) {
        this.radius = radius;
    }

    @Override
    public double area() {
        return PI * radius * radius;
    }

    @Override
    public double perimeter() {
        return 2 * PI * radius;
    }

    @Override
    public void draw_shape() {
```

```
        System.out.println("Drawing a Circle with radius: " + radius);
    }
}

class Rectangle implements Figure, Draw {
    double length, width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    @Override
    public double area() {
        return length * width;
    }

    @Override
    public double perimeter() {
        return 2 * (length + width);
    }

    @Override
    public void draw_shape() {
        System.out.println("Drawing a Rectangle with length: " + length + " and width:
" + width);
    }
}

public class MultipleInheritanceTest {
    public static void main(String[] args) {
        Figure figure;
        Draw draw;

        figure = new Circle(5);
        draw = (Draw) figure;
        System.out.println("Circle Area: " + figure.area());
        System.out.println("Circle Perimeter: " + figure.perimeter());
        draw.draw_shape();

        figure = new Rectangle(10, 20);
        draw = (Draw) figure;
```

```
        System.out.println("Rectangle Area: " + figure.area());
        System.out.println("Rectangle Perimeter: " + figure.perimeter());
        draw.draw_shape();
    }
}
```

```
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 % cd "/Users/shi
tipleInheritanceTest.java && java MultipleInheritanceTest
Circle Area: 78.53975
Circle Perimeter: 31.4159
Drawing a Circle with radius: 5.0
Rectangle Area: 200.0
Rectangle Perimeter: 60.0
Drawing a Rectangle with length: 10.0 and width: 20.0
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
```

Q13. WAP to demonstrate the concept of method overriding.

```
class Animal {
    void sound() {
        System.out.println("Animal makes a sound");
    }
}

class Dog extends Animal {
    @Override
    void sound() {
        System.out.println("Dog barks");
    }
}

class Cat extends Animal {
    @Override
    void sound() {
        System.out.println("Cat meows");
    }
}

public class MethodOverridingTest {
    public static void main(String[] args) {
        Animal myAnimal = new Animal();
        Animal myDog = new Dog();
        Animal myCat = new Cat();
```

```
        myAnimal.sound();
        myCat.sound();
    }
}
```

```
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 % cd "/Users/shik
hodOverridingTest.java && java MethodOverridingTest
Animal makes a sound
Cat meows
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
```

Q14. WAP to demonstrate the use of super keyword to call overridden methods and instance Variables.

```
class Animal {
    String name = "Generic Animal";

    void sound() {
        System.out.println("Animal makes a sound");
    }
}

class Dog extends Animal {
    String name = "Dog";

    @Override
    void sound() {
        System.out.println("Dog barks");
    }

    void display() {
        System.out.println("Name in Animal class: " + super.name);

        sound();
        super.sound();
    }
}

public class SuperKeywordTest {
    public static void main(String[] args) {
        Dog dog = new Dog();
```

```
        dog.display();
    }
}
```

```
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 % cd "/Use
erKeywordTest.java && java SuperKeywordTest
Name in Animal class: Generic Animal
Dog barks
Animal makes a sound
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
```

Q15. Create a class Animal with
• instance variables:
• boolean vegetarian
• String food
• int numOfLegs
• Create a no-argument constructor and parameterized constructor. (Use “this”)
• Create getters and setters
• Create a toString() method for animal class
• Create a subclass Cat with instance variable:
• String color

• Create a no-argument constructor and parameterized constructor which has all four parameters (Use this and super)
• Create a toString() method for Cat class
• Create a subclass Cow with instance variable:
• String breed
• Create a no-argument constructor and parameterized constructor which has all four parameters. (Use this and super)
• Create a toString() method for Cow class

```
class Animal {

    private boolean vegetarian;
    private String food;
    private int numOfLegs;


    public Animal() {
```

```
        this.vegetarian = false;
        this.food = "Unknown";
        this.numOfLegs = 0;
    }


    public Animal(boolean vegetarian, String food, int numOfLegs) {
        this.vegetarian = vegetarian;
        this.food = food;
        this.numOfLegs = numOfLegs;
    }

    public boolean isVegetarian() {
        return vegetarian;
    }

    public void setVegetarian(boolean vegetarian) {
        this.vegetarian = vegetarian;
    }

    public String getFood() {
        return food;
    }

    public void setFood(String food) {
        this.food = food;
    }

    public int getNumOfLegs() {
        return numOfLegs;
    }

    public void setNumOfLegs(int numOfLegs) {
        this.numOfLegs = numOfLegs;
    }


    public String toString() {
        return "Animal [Vegetarian=" + vegetarian + ", Food=" + food + ", Number of
Legs=" + numOfLegs + "]";
    }
}
```

```java
class Cat extends Animal {
    private String color;

    public Cat() {
        super();
        this.color = "Unknown";
    }

    public Cat(boolean vegetarian, String food, int numOfLegs, String color) {
        super(vegetarian, food, numOfLegs);
        this.color = color;
    }


    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }


    @Override
    public String toString() {
        return "Cat [Color=" + color + ", " + super.toString() + "]";
    }
}


class Cow extends Animal {

    private String breed;

    public Cow() {
        super();
        this.breed = "Unknown";
    }
```

```java
    public Cow(boolean vegetarian, String food, int numOfLegs, String breed) {
        super(vegetarian, food, numOfLegs);
        this.breed = breed;
    }


    public String getBreed() {
        return breed;
    }

    public void setBreed(String breed) {
        this.breed = breed;
    }

    @Override
    public String toString() {
        return "Cow [Breed=" + breed + ", " + super.toString() + "]";
    }
}

public class AnimalDemo1 {
    public static void main(String[] args) {

        Animal animal = new Animal();
        System.out.println(animal);

        Animal animal2 = new Animal(true, "Grass", 4);
        System.out.println(animal2);


        Cat cat = new Cat();
        System.out.println(cat);

        Cat cat2 = new Cat(false, "Fish", 4, "Black");
        System.out.println(cat2);


        Cow cow = new Cow();
        System.out.println(cow);

        Cow cow2 = new Cow(true, "Hay", 4, "Jersey");
        System.out.println(cow2);
```

```
    }
}
```

```
cd "/Users/shikhasingh/Desktop/java assignment 2.1%
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 % cd "/Users/shikhasingh/Desktop
malDemo1.java && java AnimalDemo1
Animal [Vegetarian=false, Food=Unknown, Number of Legs=0]
Animal [Vegetarian=true, Food=Grass, Number of Legs=4]
Cat [Color=Unknown, Animal [Vegetarian=false, Food=Unknown, Number of Legs=0]]
Cat [Color=Black, Animal [Vegetarian=false, Food=Fish, Number of Legs=4]]
Cow [Breed=Unknown, Animal [Vegetarian=false, Food=Unknown, Number of Legs=0]]
Cow [Breed=Jersey, Animal [Vegetarian=true, Food=Hay, Number of Legs=4]]
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
```

• Q16. Create a figure class with instance variable dim1, dim2 and method as area(). Create two derived class rectangle and triangle with constructors defined for initializing instance variable. Override area in both derived classes. WAP to demonstrate dynamic method dispatch (Run time polymorphism).

```
class Figure {
    protected double dim1;
    protected double dim2;

    public Figure(double dim1, double dim2) {
        this.dim1 = dim1;
        this.dim2 = dim2;
    }

    public double area() {
        System.out.println("Area is not defined for generic Figure.");
        return 0;
    }
}

class Rectangle extends Figure {
    public Rectangle(double length, double breadth) {
        super(length, breadth);
    }

    @Override
```

```
    public double area() {
        return dim1 * dim2;
    }
}

class Triangle extends Figure {
    public Triangle(double base, double height) {
        super(base, height);
    }

    @Override
    public double area() {
        return 0.5 * dim1 * dim2;
    }
}

public class FigureDemo {
    public static void main(String[] args) {
        Figure figureRef = new Rectangle(10, 20);
        System.out.println("Area of Rectangle: " + figureRef.area());

        figureRef = new Triangle(10, 5);
        System.out.println("Area of Triangle: " + figureRef.area());
    }
}
```

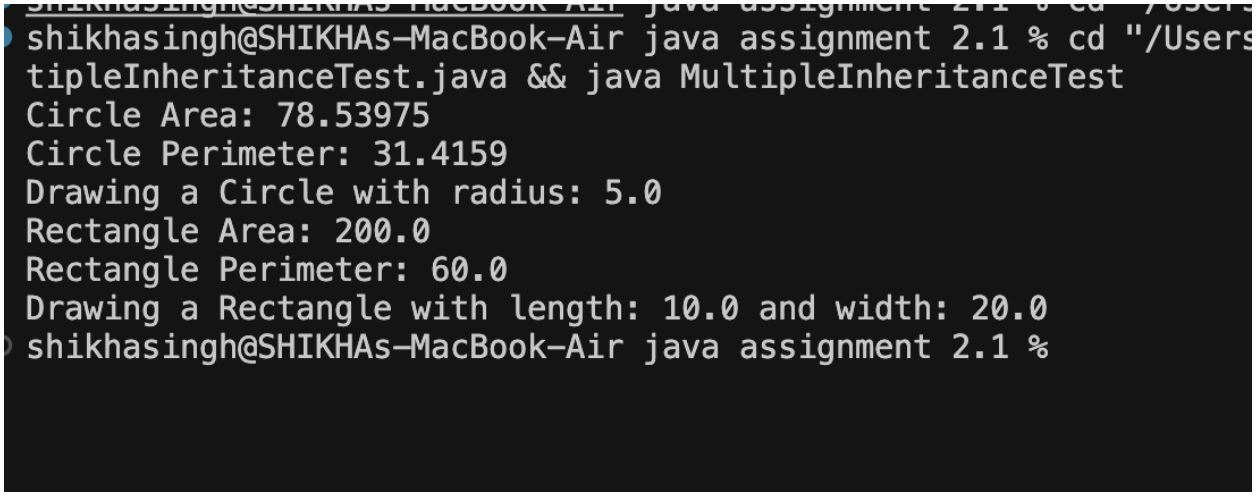
```
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 % cd
ureDemo.java && java FigureDemo
Area of Rectangle: 200.0
Area of Triangle: 25.0
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
```

• Q17. Write a program to implement the concept of multiple inheritance through interface. Create a
Figure and Draw interface. In Figure interface, define members PI, area(), perimeter(). In Draw

interface, define members draw_shape(). Implement it in Circle class and Rectangle class.

```
interface Figure {
   double PI = 3.14159;

   double area();
   double perimeter();
}

interface Draw {
   void draw_shape();
}

class Circle implements Figure, Draw {
   double radius;

   Circle(double radius) {
       this.radius = radius;
   }

   @Override
   public double area() {
       return PI * radius * radius;
   }

   @Override
   public double perimeter() {
       return 2 * PI * radius;
   }

   @Override
   public void draw_shape() {
       System.out.println("Drawing a Circle with radius: " + radius);
   }
}

class Rectangle implements Figure, Draw {
   double length, width;

   Rectangle(double length, double width) {
       this.length = length;
       this.width = width;
```

```
    }

    @Override
    public double area() {
        return length * width;
    }

    @Override
    public double perimeter() {
        return 2 * (length + width);
    }

    @Override
    public void draw_shape() {
        System.out.println("Drawing a Rectangle with length: " + length + " and width:
" + width);
    }
}

public class MultipleInheritanceTest {
    public static void main(String[] args) {
        Figure figure;
        Draw draw;

        figure = new Circle(5);
        draw = (Draw) figure;
        System.out.println("Circle Area: " + figure.area());
        System.out.println("Circle Perimeter: " + figure.perimeter());
        draw.draw_shape();

        figure = new Rectangle(10, 20);
        draw = (Draw) figure;
        System.out.println("Rectangle Area: " + figure.area());
        System.out.println("Rectangle Perimeter: " + figure.perimeter());
        draw.draw_shape();
    }
}
```



• Q18. Write a Java program that creates a class hierarchy for employees of a company. The base
class should be Employee, with subclasses Manager and Developer.
• Employee class should have private attributes such as name, address, salary, and job title. Implement public methods:
• public double calculateBonus(): return 0.0
• public String generatePerformanceReport(): return "No performance report available."
• Create getters
• Create no arg and parameterized constructors
• Manager class has an attribute numberOfSubordinates. It has a method: mangeProject(), and overridden methods: calculateBonus() and generatePerformanceReport().
• public double calculateBonus(): provides a custom implementation for bonus calculation for managers. In this case, it calculates the bonus as 15% of the manager's salary. (Hint: Use getter for salary)
• public String generatePerformanceReport(): It returns a specific performance report message for managers, including the manager's name and an "Excellent" rating. (Hint: Use getter for name)
• public void manageProject(): This is a custom method specific to the "Manager" class. It simulates the action of a manager managing a project by printing a message to the console. (Hint: Use getter for name)
• Create no arg and parameterized constructors and getters
• Developer class has a private attribute programmingLanguage

• public double calculateBonus(): provides a custom implementation for bonus calculation for developers. In this case, it calculates the bonus as 10% of the developer's salary. (Hint: Use getter for salary)
• public String generatePerformanceReport(): It returns a specific performance report message for developers, including the developer's name and a "Good" rating. (Hint:

Use getter for name)
• public void writeCode(): This is a custom method specific to the "Developer" class. It simulates the action of a developer writing code in their specialized programming language by printing a message to the console. (Hint: Use getter for name)
• Create no arg and parameterized constructors and getters

```
class Employee {
   private String name;
   private String address;
   private double salary;
   private String jobTitle;

   // No-argument constructor
   public Employee() {
       this.name = "";
       this.address = "";
       this.salary = 0.0;
       this.jobTitle = "";
   }

   // Parameterized constructor
   public Employee(String name, String address, double salary, String jobTitle) {
       this.name = name;
       this.address = address;
       this.salary = salary;
       this.jobTitle = jobTitle;
   }

   // Getters
   public String getName() {
       return name;
   }

   public String getAddress() {
       return address;
   }

   public double getSalary() {
       return salary;
   }

   public String getJobTitle() {
```

```
        return jobTitle;
    }

    // Method to calculate bonus
    public double calculateBonus() {
        return 0.0;
    }

    // Method to generate performance report
    public String generatePerformanceReport() {
        return "No performance report available.";
    }
}

class Manager extends Employee {
    private int numberOfSubordinates;

    // No-argument constructor
    public Manager() {
        super();
        this.numberOfSubordinates = 0;
    }

    // Parameterized constructor
    public Manager(String name, String address, double salary, String jobTitle, int
numberOfSubordinates) {
        super(name, address, salary, jobTitle);
        this.numberOfSubordinates = numberOfSubordinates;
    }

    // Getter for numberOfSubordinates
    public int getNumberOfSubordinates() {
        return numberOfSubordinates;
    }

    // Method to manage project
    public void manageProject() {
        System.out.println(getName() + " is managing a project.");
    }

    // Overridden method to calculate bonus
    @Override
```

```
    public double calculateBonus() {
        return 0.15 * getSalary();
    }

    // Overridden method to generate performance report
    @Override
    public String generatePerformanceReport() {
        return "Manager " + getName() + " has an Excellent rating.";
    }
}

class Developer extends Employee {
    private String programmingLanguage;

    // No-argument constructor
    public Developer() {
        super();
        this.programmingLanguage = "";
    }

    // Parameterized constructor
    public Developer(String name, String address, double salary, String jobTitle,
String programmingLanguage) {
        super(name, address, salary, jobTitle);
        this.programmingLanguage = programmingLanguage;
    }

    // Getter for programmingLanguage
    public String getProgrammingLanguage() {
        return programmingLanguage;
    }

    // Method to write code
    public void writeCode() {
        System.out.println(getName() + " is writing code in " + programmingLanguage +
".");
    }

    // Overridden method to calculate bonus
    @Override
    public double calculateBonus() {
        return 0.10 * getSalary();
```

```
    }

    // Overridden method to generate performance report
    @Override
    public String generatePerformanceReport() {
        return "Developer " + getName() + " has a Good rating.";
    }
}

public class EmployeeTest {
    public static void main(String[] args) {
        Manager manager = new Manager("Ankita", "123 Manager St", 80000, "Manager", 5);
        Developer developer = new Developer("shikha", "456 Developer Ave", 60000,
"Developer", "Java");

        System.out.println("Manager Details:");
        System.out.println("Name: " + manager.getName());
        System.out.println("Address: " + manager.getAddress());
        System.out.println("Salary: " + manager.getSalary());
        System.out.println("Job Title: " + manager.getJobTitle());
        System.out.println("Number of Subordinates: " +
manager.getNumberOfSubordinates());
        System.out.println("Bonus: " + manager.calculateBonus());
        System.out.println("Performance Report: " +
manager.generatePerformanceReport());
        manager.manageProject();

        System.out.println("\nDeveloper Details:");
        System.out.println("Name: " + developer.getName());
        System.out.println("Address: " + developer.getAddress());
        System.out.println("Salary: " + developer.getSalary());
        System.out.println("Job Title: " + developer.getJobTitle());
        System.out.println("Programming Language: " +
developer.getProgrammingLanguage());
        System.out.println("Bonus: " + developer.calculateBonus());
        System.out.println("Performance Report: " +
developer.generatePerformanceReport());
        developer.writeCode();
    }
}
```

```
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 % cd "/Users/shikhasingh/Desktop/java assignment 2.1/" && javac Emp
loyeeTest.java && java EmployeeTest
Manager Details:
Name: Ankita
Address: 123 Manager St
Salary: 80000.0
Job Title: Manager
Number of Subordinates: 5
Bonus: 12000.0
Performance Report: Manager Ankita has an Excellent rating.
Ankita is managing a project.

Developer Details:
Name: shikha
Address: 456 Developer Ave
Salary: 60000.0
Job Title: Developer
Programming Language: Java
Bonus: 6000.0
Performance Report: Developer shikha has a Good rating.
shikha is writing code in Java.
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
```

Q19. Write a Java program to create an abstract class Animal with an abstract method called sound().

```java
abstract class Animal {

    abstract void sound();
}

class Dog extends Animal {
    @Override
    void sound() {
        System.out.println("Dog barks");
    }
}

class Cat extends Animal {
    @Override
    void sound() {
        System.out.println("Cat meows");
    }
}

public class AbstractClassTest {
    public static void main(String[] args) {
        Animal myDog = new Dog();
        Animal myCat = new Cat();

        myDog.sound();
        myCat.sound();
    }
}
```

```
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 % cd "/Users/
tractClassTest.java && java AbstractClassTest
Dog barks
Cat meows
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
```

Q20. Create subclasses Lion and Cat that extend the Animal class and implement the sound() method to make a specific sound for each animal.

```java
abstract class Animal {
    private String name;

    public Animal(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public abstract void sound();
}

class Lion extends Animal {

    public Lion(String name) {
        super(name);
    }

    @Override
    public void sound() {
        System.out.println(getName() + " roars!");
    }
}

class Cat extends Animal {

    public Cat(String name) {
        super(name);
```

```
    }

    @Override
    public void sound() {
        System.out.println(getName() + " meows!");
    }
}

public class LionDemo {
    public static void main(String[] args) {
        Animal lion = new Lion("Lion");
        lion.sound();

        Animal cat = new Cat("Cat");
        cat.sound();
    }
}
```

```
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 % cd "/Users/shikhasingh/De
nDemo.java && java LionDemo
Lion roars!
Cat meows!
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
```

Q21. Create an Abstract Class “Shape” and 2 subclasses, SemiCircle and Circle that extend the Shape class and implement the respective methods to calculate the area and perimeter of each shape.

• Shape also defines 2 abstract methods calculateArea() and calculateCircumference(). Both return doubles representing the area and circumference of the shapes respectively.

• For each shape you will have to create the necessary values need to calculate area and circumference and getters and setters for each of the values. You will also need to create constructors for each class, and the instructions for those will be included with each class.

```
abstract class Shape {
    abstract double calculateArea();
    abstract double calculateCircumference();
}

class Circle extends Shape {
    private double radius;

    // No-argument constructor
    public Circle() {
```

```
        this.radius = 0.0;
    }

    // Parameterized constructor
    public Circle(double radius) {
        this.radius = radius;
    }

    // Getter for radius
    public double getRadius() {
        return radius;
    }

    // Setter for radius
    public void setRadius(double radius) {
        this.radius = radius;
    }

    @Override
    double calculateArea() {
        return Math.PI * radius * radius;
    }

    @Override
    double calculateCircumference() {
        return 2 * Math.PI * radius;
    }
}

class SemiCircle extends Shape {
    private double radius;

    // No-argument constructor
    public SemiCircle() {
        this.radius = 0.0;
    }

    // Parameterized constructor
    public SemiCircle(double radius) {
        this.radius = radius;
    }
```

```java
    // Getter for radius
    public double getRadius() {
        return radius;
    }

    // Setter for radius
    public void setRadius(double radius) {
        this.radius = radius;
    }

    @Override
    double calculateArea() {
        return 0.5 * Math.PI * radius * radius;
    }

    @Override
    double calculateCircumference() {
        return Math.PI * radius + 2 * radius;
    }
}

public class ShapeTest {
    public static void main(String[] args) {
        Circle circle = new Circle(5.0);
        SemiCircle semiCircle = new SemiCircle(5.0);

        System.out.println("Circle with radius " + circle.getRadius() + ":");
        System.out.println("Area: " + circle.calculateArea());
        System.out.println("Circumference: " + circle.calculateCircumference());

        System.out.println("\nSemiCircle with radius " + semiCircle.getRadius() + ":");
        System.out.println("Area: " + semiCircle.calculateArea());
        System.out.println("Circumference: " + semiCircle.calculateCircumference());
    }
}
```

```
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 % cd "/Users/shikhasingh/Des
peTest.java && java ShapeTest
Circle with radius 5.0:
Area: 78.53981633974483
Circumference: 31.41592653589793

SemiCircle with radius 5.0:
Area: 39.269908169872416
Circumference: 25.707963267948966
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
```

Q22. Write a program to search an element in an array using for each loop.

```java
public class ArraySearch {
    public static void main(String[] args) {
        int[] numbers = {10, 20, 30, 40, 50};
        int target = 30;
        boolean found = false;


        for (int number : numbers) {
            if (number == target) {
                found = true;
                break;
            }
        }

        if (found) {
            System.out.println("Element " + target + " is present in the array.");
        } else {
            System.out.println("Element " + target + " is not present in the array.");
        }
    }
}
```

```
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
aySearch.java && java ArraySearch
Element 30 is present in the array.
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
```

Q23. WAP to calculate sum and average of elements stored in one D array.

```java
public class ArraySumAverage {
    public static void main(String[] args) {
        int[] numbers = {10, 20, 30, 40, 50};
        int sum = 0;
        double average;


        for (int number : numbers) {
            sum += number;
        }


        average = (double) sum / numbers.length;


        System.out.println("Sum of elements: " + sum);
        System.out.println("Average of elements: " + average);
    }
}
```

```
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 % cd "/Users/sh
aySumAverage.java && java ArraySumAverage
Sum of elements: 150
Average of elements: 30.0
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
```

Q24. WAP to input elements in one D array using scanner class.

```java
import java.util.Scanner;

public class ArrayInput {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);


        System.out.print("Enter the number of elements: ");
        int n = scanner.nextInt();


        int[] numbers = new int[n];


        System.out.println("Enter " + n + " elements:");
```

```
        for (int i = 0; i < n; i++) {
            System.out.print("Element " + (i + 1) + ": ");
            numbers[i] = scanner.nextInt();
        }



        scanner.close();



        System.out.println("Array elements are:");
        for (int number : numbers) {
            System.out.println(number);
        }
    }
}
```

```
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 % cd "/Users/shikhasingh/D
ayInput.java && java ArrayInput
Enter the number of elements: 4
Enter 4 elements:
Element 1: 23
Element 2: 23
Element 3: 45
Element 4: 67
Array elements are:
23
23
45
67
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 % $
```

Q25. Write a Java program to find the maximum and minimum value of an array.

```
public class ArrayMinMax {
    public static void main(String[] args) {
        int[] numbers = {10, 20, 30, 5, 15, 25};



        int min = numbers[0];
        int max = numbers[0];



        for (int number : numbers) {
            if (number < min) {
                min = number;
```

```
            }
            if (number > max) {
                max = number;
            }
        }

        System.out.println("Minimum value: " + min);
        System.out.println("Maximum value: " + max);
    }
}
```

```
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 % cd "/Users/shikhasingh/
ayMinMax.java && java ArrayMinMax
Minimum value: 5
Maximum value: 30
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
```

Q26. Write a Java program to sort an array of integers.

```
import java.util.Arrays;

public class ArraySort {
    public static void main(String[] args) {
        int[] numbers = {40, 10, 30, 20, 50};

        System.out.println("Original array:");
        for (int number : numbers) {
            System.out.print(number + " ");
        }
        System.out.println();

        Arrays.sort(numbers);


        System.out.println("Sorted array:");
        for (int number : numbers) {
            System.out.print(number + " ");
        }
    }
}
```

```
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 % cd "/Users/shikhasingh/Desktop/java assignment 2.1/" &&
aySort.java && java ArraySort
Original array:
40 10 30 20 50
Sorted array:
10 20 30 40 50 %
shikhasingh@SHIKHAs-MacBook-Air java assignment 2.1 %
```